Operations Conductor

Implementation Guide

Operations Conductor: Implementation Guide

Table of Contents

Automate manual tasks and processes with Operations Conductor

Publication date: November 2019 (**last update (p. 38): December 2021**)

This implementation guide discusses architectural considerations and configuration steps for deploying Operations Conductor in the Amazon Web Services (AWS) Cloud. It includes a link to an AWS CloudFormation template that launches and configures the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects, administrators, and DevOps professionals who have practical experience architecting on the AWS Cloud.

Overview

Amazon Web Services (AWS) offers its customers several methods to help automate manual tasks or processes such as archiving, data backup and recovery, resource selection and scheduling, and configuration management. Automating these tasks can help increase reliability, reduce costs, and reduce operational complexity so you can focus on delivering applications and services at a high velocity. However, it can be a challenge to automate operational tasks on various AWS resources across multiple regions and accounts.

To help customers reduce operational complexity, AWS offers Operations Conductor, a solution that creates a simple web interface for automating and orchestrating operational tasks. The solution enables administrators to quickly create manual, event-based or time-based triggers for managing resources.

The solution deploys a set of common operational actions that can be configured to automate administration tasks, resource scaling, and cost management. The solution can be customized and extended to fit your business needs.

Cost

You are responsible for the cost of the AWS services used while running this reference deployment. The total cost for running this solution depends on the number of tasks you run. As of the date of publication, the cost for running this solution with default settings in the US East (N. Virginia) Region is approximately **$5 per month**. This cost estimate assumes 1GB of Amazon DynamoDB data storage, 200,000 DynamoDB write capacity units at the On-Demand capacity mode, 200,000 DynamoDB read capacity units at the On-Demand capacity mode, 1GB of Amazon CloudWatch Logs ingested, 1 million Amazon CloudWatch custom events generated, 1 million AWS Lambda function invocations, and 100,000 AWS System Manager automations.

This cost estimate does not reflect variable charges incurred from the executed tasks, data transfer, or snapshot storage. Prices are subject to change. For full details, see the pricing webpage for each AWS service you will be using in this solution.

Architecture Overview

Deploying this solution builds the following environment in the AWS Cloud.

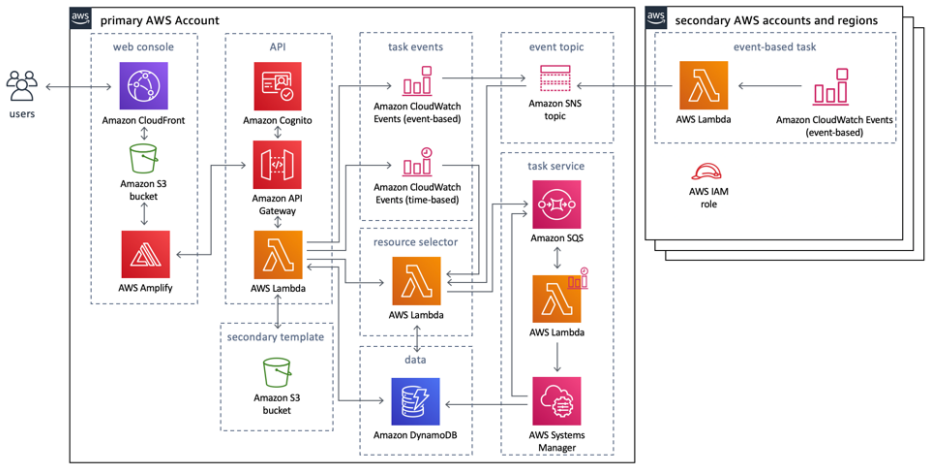

Figure 1: Operations Conductor architecture on AWS

This solution includes an AWS CloudFormation template that you deploy in the primary account. This template launches an Amazon API Gateway to invoke the solution's microservices (AWS Lambda functions). The microservices provide the business logic to manage events and tasks. The microservices interact with Amazon Simple Queue Service (Amazon SQS), AWS Systems Manager, and Amazon DynamoDB to provide storage for task details and results.

The primary template also automatically generates additional AWS CloudFormation templates in an Amazon Simple Storage Service (Amazon S3) bucket. These templates enable you to create cross-account and region AWS Identity and Access Management (IAM) roles to perform actions in secondary accounts and regions, and forward events. You can modify and build upon these templates to create custom actions that extend the solution's functionality.

When a user creates a task, the `task service` Lambda function creates an Amazon CloudWatch Events rule and generates an AWS CloudFormation template that is stored in the solution-created `cloudformation template` Amazon S3 bucket. This template must then be deployed by the user in all accounts and regions where the solution should execute that task. Deploying a secondary template creates an IAM role and policy to grant the solution the necessary permissions required to act on resources as the task is executed. When tasks are event-based, the secondary template also deploys an Amazon CloudWatch Events rule to intake resource events, and a Lambda function to forward the event to the Amazon Simple Notification Service (Amazon SNS) topic.

Solution Components

Actions

The Operations Conductor solution deploys a set of pre-defined actions that can be performed on AWS resources. The business logic for performing actions on resources is stored in an AWS Systems Manager automation document that the solution creates when it launches.

When the template is deployed, a `custom resource` AWS Lambda function creates documents with the tag name and value provided in the **Document Tag Key** and **Document Tag Value** AWS CloudFormation template parameters. For a complete list of pre-installed actions, see Appendix B (p. 19). You can add additional actions by creating your own documents using the same tag name and value. For more information about customizing actions, see Appendix C (p. 23).

Tasks

A task represents configuring an action to be performed on a set of resources. When creating a task, you supply a tag key that is used to identify the group of resources on which this action will be performed. Tasks are executed on a fixed schedule, in response to an event on an individual resource, or manually triggered in the solution's web console. Each task can only have one action enabled. For more information on task configuration, see Appendix A (p. 17).

Scheduled Tasks

When defining your schedule, you can supply a cron or rate expression. For more information, see Schedule Expressions for Rules in the Amazon CloudWatch Events User Guide.

Event-Based Tasks

Operations Conductor will perform the action on the resource that triggered an Amazon CloudWatch Events rule. For more information about acceptable event patterns, see Event Patterns in CloudWatch Events in the *Amazon CloudWatch Events User Guide*.

API Microservices

User Microservice

The user microservice manages users in the Amazon Cognito user pool. Only users who belong to the admin group can access the microservice, add users, edit user groups, and delete users.

Action Microservice

The action microservice tags AWS Systems Manager documents with values provided in the **Document Tag Key** and **Document Tag Value** template parameters. The microservice also identifies documents by

the **Document Tag Key** and **Document Tag Value**, which enables users to add documents and extend the solution's functionality.

Task Microservice

The task microservice creates tasks, shows created tasks, edits tasks, deletes tasks, and manually executes tasks. When a task is created, an AWS CloudFormation template andAmazon CloudWatch Events rule are automatically created for secondary accounts and regions.

Resource Selector

The `resource selector` AWS Lambda function identifies resources and queues them until the actions are performed on the resources. Queueing individual resources enables actions to be performed at a large scale. The solution also employs automatic retry logic to retry failed executions due to service limits.

The Lambda function is triggered when a task is executed manually using the solution's web console, or automatically in response to the event or schedule-based trigger configured for the task. When the function is triggered, it uses the Resource Groups Tagging API to find resources tagged with the `Target Tag` that was supplied when the task was created. Resources are then filtered by the resource type defined in the AWS Systems Manager automation document. A message is placed in the solution's `resource queue` for each resource that was identified.

Queue Consumer

The `queue consumer` AWS Lambda function is triggered by an Amazon CloudWatch Events rule and reads batches of messages in the `resource` queue on a fixed schedule (every 60 seconds by default). The function then executes the correct Systems Manager automation document for the action and sets the required input parameters for each message in the queue.

The `queue consumer` calls the `DescribeAutomationExecutions` AWS Systems Manager API to identify how many automation executions are currently running. Messages will be read in batches only when the Systems Manager concurrent running automations are not full. See AWS Service Quotas in the *AWS General Reference guide*.

AWS Systems Manager

Operations Conductor uses an AWS Systems Manager automation document to store the business logic for performing each action on a resource. For more information on creating automation documents, see Appendix C (p. 23).

Amazon DynamoDB

Task Table

When you deploy the AWS CloudFormation template, the solution creates an Amazon DynamoDB tasks table. The following table shows the descriptions and item attributes in the task table.

Attribute	Type	Description
taskId (Partition Key)	String	The UUID of the task
actionName	String	The action name of the task
name	String	The task name
description	String	The task description
enabled	Boolean	Flag indicating whether the automated task executions are enabled
Target Tag	String	Target tag which will select the task resource(s) to execute the automation
Task Parameters	List	Parameters which will be used by AWS Systems Manager document
Trigger Type	String	The task trigger type: `Schedule` or `Event`
Template URL	String	The location of the secondary template that contains a role and policy allowing the solution to perform the action.
Scheduled Type	String	The scheduled type: `CronExepression` or `FixedRate`
Scheduled Cron Expression	String	The cron expression of the task. This attribute is available if you selected `schedule` for **Trigger Type**, and `scheduledType` for **Cron Expression**.
Scheduled Fixed Rate Interval	Number	The scheduled fixed rate interval. This attribute is available if you selected `schedule` for **Trigger Type**, and `scheduledType` for **Fixed Rate**.
Scheduled Fixed Rate Type	String	The scheduled fixed rate type. This attribute is available if you selected `schedule` for **Trigger Type**, and `scheduledType` for **Fixed Rate**.
Event Pattern	String	The event pattern of the task execution. This attribute is available if you selected `triggerType` for **Event Pattern**
accounts	List <String>	The accounts where tasks will be executed

Attribute	Type	Description
regions	List <String>	The regions where tasks will be executed

Task Executions Table

The items task executions table contain the following attributes.

Attribute	Type	Description
taskId (Partition Key)	String	The UUID of the task
parentExecutionId (Sort Key)	String	The task execution ID
status	String	The task status
totalResourceCount	Number	The total resource count of the task
completedResourceCount	Number	The completed resource count of the task
startTime	String	The task start time

Task Executions Table Index

The task execution IDs table contains an index which enables users to sort task executions by start time.

Attribute	Type	Description
taskId (Partition Key)	String	The UUID of the task
startTime (Sort Key)	String	The task start time
parentExecutionId	String	The task execution ID
status	Number	The task status
totalResourceCount	Number	The total resource count of the task
completedResourceCount	Number	The tcompleted resource count of the task

Automation Executions Table

The task execution IDs are mapped to the task IDs in theAmazon DynamoDB `automation executions` table.

Attribute	Type	Description
parentExecutionId (Partition Key)	String	The task execution ID
automationExecutionId (Sort Key)	String	The automation execution ID

Considerations

Amazon Cognito Limits

Operations Conductor uses an Amazon Cognito user pool to manage users. Amazon Cognito sends an email every time you create a user, change a password, or reset a password. Amazon Cognito limits the number of emails sent daily per user pool to 50. For customers who plan to use this solution for a large number of users, we recommend using Amazon Simple Email Service (Amazon SES) for these emails. For more information, see Authorizing Amazon Cognito to Send Amazon SES Email on Your Behalf in the *Amazon Cognito Developer Guide*.

Copying an Amazon EBS Snapshot

Operations Conductor provides copying snapshots to other regions and other accounts. Each account can have up to five concurrent snapshot copy requests to a single destination region. For more information, see Copying an Amazon EBS Snapshot in the *Amazon Elastic Compute Cloud User Guide*.

Regional Deployments

Operations Conductor uses Amazon Cognito which is currently available in specific AWS regions only. Therefore, you must launch this solution in a region where Amazon Cognito is available. For the most current service availability by region, see AWS service offerings by region.

AWS CloudFormation Template

This solution uses AWS CloudFormation to automate the deployment of the Operations Conductor solution in the AWS Cloud. It includes the following AWS CloudFormation template, which you can download before deployment:

operations-conductor.template: Use this template to launch the solution and all associated components. The default configuration deploys AWS Lambda, Amazon API Gateway, Amazon Simple Storage Service, Amazon CloudFront, Amazon Cognito, Amazon DynamoDB, Amazon Simple Notification Service, Amazon Simple Queue Service, Amazon CloudWatch Events, and AWS Systems Manager, but you can also customize the template based on your specific needs.

Automated Deployment

Before you launch the automated deployment, please review the architecture and considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy Operations Conductor into your account.

Time to deploy: Approximately 30 minutes

What We'll Cover

The procedure for deploying this architecture on AWS consists of the following steps. For detailed instructions, follow the links for each step.

Step 1. Launch the Stack in Your Primary Account (p. 11)

- Launch the AWS CloudFormation template into your AWS account.
- Enter values for required parameters: **Stack Name**, **Document Tag Key**, **Document Tag Value**, and **Administrator Email**.
- Review the other template parameters, and adjust if necessary.

Step 2. Create Tasks in the Web Console (p. 13)

- Log in to the console and create tasks.

Step 3. Launch the Template in the Secondary Account(s) (p. 13)

- Copy the AWS CloudFormation template URL.
- Launch the template into secondary account(s) and region(s).

Step 4. Tag Your Resources (p. 13)

- Apply the target tag to applicable resources.

Step 1. Launch the Stack in Your Primary Account

This automated AWS CloudFormation template deploys Operations Conductor in the AWS Cloud.

> **Note**
> You are responsible for the cost of the AWS services used while running this solution. See the Cost (p. 2) section for more details. For full details, see the pricing webpage for each AWS service you will be using in this solution.

1. Sign in to the AWS Management Console and click the button below to launch the `operations-conductor` AWS CloudFormation template.

You can also download the template as a starting point for your own implementation.

2. The template is launched in the US East (N. Virginia) Region by default. To launch this solution in a different AWS Region, use the region selector in the console navigation bar.

   Note
   Operations Conductor uses Amazon Cognito which is currently available in specific AWS regions only. Therefore, you must launch this solution in a region where Amazon Cognito is available. For the most current service availability by region, see AWS service offerings by region.

3. On the **Create stack** page, verify that the correct template URL shows in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify Details** page, assign a name to your solution stack.
5. Under **Parameters**, review the parameters for the template, and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
Document Tag Key	*<Requires Input>*	The Systems Manager automation document tag key used to filter the solution's automation documents
Document Tag Value	*<Requires Input>*	The Systems Manager automation document tag value which will be used to filter the operations conductor automation documents
Administrator Name	`Administrator`	The username for the administrator
Administrator Email	*<Requires Input>*	The email address of the administrator for the web console. After launch, an email will be sent to the address with a temporary password for the web console.
Log Level	`2`	The microservices log level. For more information, see Appendix D (p. 32).

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation console in the Status column. You should see a status of **CREATE_COMPLETE** in approximately 30 minutes.

Note
In addition to the primary Lambda functions, this solution includes the `Custom Resource` Lambda function, which runs only during initial configuration or when resources are updated or deleted. When running this solution, the `Custom Resource` function is inactive. However, do not delete the `Custom Resource` function as it is necessary to manage associated resources.

Step 2. Create Tasks in the Web Console

After the AWS CloudFormation template is successfully deployed, you will receive an email containing the web console URL and temporary log in credentials.

1. After logging in to the web console, select **Get Started** to create a task.
2. In the **Action Catalog**, select an action, then select **Create Task**.
3. Begin configuring tasks.
4. Once you reviewed the tasks configuration, select **Create**.

   For more information on the included task parameters, see Appendix A (p. 17).

Step 3. Launch the Template in the Secondary Account(s)

Use this procedure to launch the template generated in the secondary account(s) and region(s) you configured when creating a task. You must deploy the template in each account and region. For example, if you create a task in two accounts and two regions, you will need to launch the template four times.

1. After you configure a task, navigate to the **task detail** page in the web console.
2. Find the AWS CloudFormation template URL, and select **Copy URL**
3. In the AWS CloudFormation console, select **Create Stack**.
4. Select specify an **Amazon S3 template URL**.
5. Paste the template link into the text box, and select **Next**.
6. Enter the **Stack Name**.
7. Select**Next**.
8. Select **Next**. Then, on the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

Step 4. Tag Your Resources

When creating tasks, every task must provide a `Target Tag`. To avoid unexpected executions or execution failures, we recommend setting a different target tag for each task unless you are using multiple schedule-based tasks with the same action. When tasks are executed, the action is performed on the resources identified by the `Target Tag`. Verify that all the resources are tagged that you want the solution to act on. For more information, see Appendix A (p. 17).

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model can reduce your operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the services operate. For more information about security on AWS, visit the AWS Security Center.

IAM Roles

AWS Identity and Access Management (IAM) roles enable customers to assign granular access policies and permissions to services and users on the AWS Cloud. The solution creates IAM roles including roles that grant the solution's AWS Lambda functions access to the other AWS services used in this solution. In addition to Lambda function roles, the primary account will assume roles to control resources in secondary accounts or regions, and the roles will be provisioned by the secondary AWS CloudFormation template.

HTTP Security Headers

HTTP security headers for the solution's web console are configured with the `Lambda@Edge` function. The included HTTP security headers are Strict-Transport-Security, Content-Security-Policy, X-Content-Type-Options, X-Frame-Options, X-XSS-Protection, and Referrer-Policy. To customize HTTP security headers, you can modify *<your-stack-name>*`-Lambda-Edge` AWS Lambda function in the N. Virginia region.

When you delete the solution stack, you must delete the `Lambda@Edge` function manually. For more information, see Deleting Lambda@Edge Functions and Replicas in the *Amazon CloudFront Developer Guide*.

Additional Resources

AWS services

- AWS CloudFormation
- AWS Identity and Access Management
- Amazon API Gateway
- AWS Lambda
- Amazon Cognito
- Amazon Simple Storage Service
- Amazon CloudFront
- Amazon DynamoDB
- Amazon Simple Queue Service
- Amazon CloudWatch Events
- Amazon Simple Notification Service
- AWS Systems Manager

Update the stack

If you have previously deployed the solution, follow this procedure to update the `operations-conductor.template` CloudFormation stack to get the latest version of the solution's framework.

1. Sign in to the AWS CloudFormation Console, select the existing Operations Conductor AWS CloudFormation stack, and select **Update**.
2. Select **Replace current template**.
3. Under **Specify template**:
    a. Select **Amazon S3 URL**.
    b. Copy the link of the latest template for the stack.
    c. Paste the link in the **Amazon S3 URL** box.
    d. Verify that the correct template URL shows in the **Amazon S3 URL** text box, and choose **Next**.
    e. Choose **Next** again.
4. Under **Parameters**, review the parameters for the template and modify them as necessary. Refer to Step 1. Launch the Stack (p. 11) for details about the parameters.
5. Choose **Next**.
6. On the **Configure stack options** page, choose **Next**.
7. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template might create AWS Identity and Access Management (IAM) resources.
8. Choose **View change set** and verify the changes.
9. Choose **Update stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should see a status of **UPDATE_COMPLETE** in approximately 30 minutes depending on the options chosen.

Appendix A: Task Configuration

When creating tasks every task must have a target tag. To avoid unexpected executions or execution failures, we recommend setting a different target tag for each task unless you are using schedule-based tasks with the same action. For example, you can set multiple tasks to set the Amazon DynamoDB capacity for tables with the same target tag but a different schedules so that tables can be scaled up and down regularly.

The following table shows the task parameters.

Parameter	Required	Description
Task Name	`Yes`	Name of the task
Task Description	`No`	Description of the task. For example, `Create a snapshot every 30 minutes`.
Target Tag	`Yes`	Target tag that will select the task resource(s) to execute the automation
Parameters	`Yes`	Parameters which will be used by the AWS Systems Manager document
Trigger Type	`Yes`	Select the task trigger type. Available trigger types: `schedule type`, `event type`.
Cron Expression	`Yes`	Select `Yes`, to set up a schedule cron expression. For more information, see Schedule Expressions for Rules in the *Amazon CloudWatch Events User Guide*.
Fixed Rate of Schedule Type	`Yes`	Select `Yes`, to set up a schedule interval. For more information, see Schedule Expressions for Rules in the *Amazon CloudWatch Events User Guide*.
Event Pattern	`Yes`	If you select `event type` for the **Trigger Type** parameter. Select `Yes`, to set up an event pattern to trigger task execution. For more information, see Schedule Expressions for Rules in the *Amazon CloudWatch Events User Guide*.
Accounts	`Yes`	Comma-delimited list of account where the task will be created.

Parameter	Required	Description
Regions	`Yes`	List of regions where the task will run. For example, `us-east-1`, `eu-west-1`.

Define a Task

Provide the **Task Name** and **Task Description**. Note that a unique task ID will be automatically generated when the task is created. If you don't provide a task name, the web console will return an error message.

Target Tag

For **Target Tag**, enter the Tag Key name that will be used by the `resource selector (p. 5)` AWS Lambda function to determine which resources need to be executed by AWS Systems Manager.

When you provide the target tag, verify that you tag your resources with the correct target tag, so the action is not taken on unexpected resources. For more information about which resources you need to tag for each action, see Appendix B (p. 19). If you don't provide a target tag, the web console will return an error message.

Parameters

Enter the values for all required and any optional Systems Manager automation document parameters. For more information about task parameters, see Appendix B (p. 19).

Task Trigger

Select the task trigger. The solution supports a `schedule type` and `event type` trigger. The schedule type supports cron and rate expressions, and the event type requires an event pattern.

Task Scope

Enter the AWS account IDs and regions where you want tasks to be executed. If you want to execute the tasks in multiple accounts and regions, use a comma separated list. To view available region codes, see Regions and Availability Zones in the *Amazon Elastic Compute Cloud User Guide*.

> **Note**
> Verify that you have launched the secondary AWS CloudFormation template in the accounts and regions you provided. For more information, see Step 3 (p. 13).

Review

Review the task parameters, and select **Create**. The API service verifies the configuration parameters, and returns an error message if any inputs are invalid.

Appendix B: Available Actions

Amazon EC2 Create Snapshot

The create snapshot action enables the solution to automatically create snapshots of Amazon Elastic Block Store (Amazon EBS) volumes. The minimum interval between task executions is 60 minutes.

Review the action-specific parameters for the template and modify them as necessary.

Parameter	Default	Description
Copy Volume Tags	`false`	Enter a tag filter to copy tags from the volume to the snapshot. For example, enter * to copy all tags from the volume to the snapshot.

Create Snapshot Action

The `resource selector` AWS Lambda function searches the Amazon EBS volumes for tagged resources with the `Target Tag` in each region and account. When you create an event task using the create snapshot action, you must set the event pattern, which will send Amazon EBS volumes as resources.

The following example JSON shows the action when an Amazon EBS volume is created.

```
{
  "source": [
    "aws.ec2"
  ],
  "detail-type": [
    "EBS Volume Notification"
  ],
  "detail": {
    "event": [
      "createVolume"
    ]
  }
}
```

Amazon EC2 Delete Snapshot

The delete snapshot action enables the solution to automatically delete snapshots of Amazon Elastic Block Store (Amazon EBS) volumes older than a customer-defined number of days. Or, customers can configure this action to keep only the latest snapshots. The minimum interval between task executions is 15 minutes.

Review the action-specific parameters for the template and modify them as necessary. This action template uses the following default values.

Parameter	Required	Description
Retention Count	`No`	The number of snapshots to retain for the EBS volume. **Note** If the total number of snapshots to retain for this volume exceeds this number, the older snapshots will be deleted.
Retention Days	`No`	The retention period in days. **Note** Snapshots older than the inputted number of days from when the task is executed will be deleted.

Delete Snapshot Action

The `resource selector` AWS Lambda function, searches the Amazon EBS volumes for tagged resources with the `Target Tag` in each region and account. When you create an event task using the create snapshot action, you must set the event pattern, which will send Amazon EBS volumes as resources.

The following example JSON shows the action when an Amazon EBS volume is modified.

```
{
  "source": [
    "aws.ec2"
  ],
  "detail-type": [
    "EBS Volume Notification"
  ],
  "detail": {
    "event": [
      "modifyVolume"
    ]
  }
}
```

Amazon EC2 Copy Snapshot

The copy snapshot action enables the solution to automatically copy snapshots of Amazon Elastic Block Store (Amazon EBS) volumes between accounts and regions. The minimum interval between task executions is 60 minutes. The maximum number of snapshots you can copy per account to a destination region concurrently is 5.

All source snapshots must be tagged with the `Target Tag` that you defined in the **Document Tag Key** template parameter. During the task creation, review the following parameters and modify them as necessary.

Parameter	Required	Description
Destination Account	`Yes`	The account where the snapshot will be copied to
Destination Region	`Yes`	The region where the snapshot will be copied

Copy Snapshot Action

The `resource selector` AWS Lambda function searches Amazon EBS volumes for tagged resources with the `Target Tag` in each region and account. When you create an event task using the create snapshot action, you must set the event pattern, which will send Amazon EBS volumes as resources.

The following example JSON shows the action when an Amazon EBS snapshot is created.

```
{
  "source": [
    "aws.ec2"
  ],
  "detail-type": [
    "EBS Snapshot Notification"
  ],
  "detail": {
    "event": [
      "createSnapshot"
    ]
  }
}
```

Amazon EC2 Resize Instance

This action stops your existing instance, resizes the instance to the next defined size up or the next defined size down, then starts the instance again.

During the resizing, Amazon Elastic Block Store (Amazon EBS) volumes on the instance will remain attached and the data will persist. However, any data on the ephemeral storage of your instance will be lost. To keep your data, it must be stored on attached Amazon EBS volumes.

Review the action-specific parameters for the template and modify them as necessary. This action template uses the following default values.

Parameter	Required	Description
Target EC2 Instance Type List	`Yes`	A comma-delimited list of instance types within the same family. The instance types should be in increasing order of size. For example, smallest instance type should be the first item in the list.

Parameter	Required	Description
CPU Utilization Low Threshold	`Yes`	An integer between 1 and 100. If the average CPU utilization goes below this threshold, the instance will be scaled down.
CPU Utilization High Threshold	`Yes`	An integer between 1 and 100. If the average CPU utilization goes above this threshold, the instance will be scaled up.

Resize Instance Action

The `resource selector` AWS Lambda function searches Amazon EC2 instances in each region and account. However, there is no recommendable event pattern for this action.

Amazon DynamoDB Set Capacity

The Amazon DynamoDB set capacity action enables the solution to automatically provision throughput capacity for reads and writes to DynamoDB tables. The minimum interval between task executions is 15 minutes.

Review the action-specific parameters for the template and modify them as necessary. This action template uses the following default values.

Parameter	Required	Description
Table Read Capacity Units	`No`	The read capacity for the table
Table Write Capacity Units	`No`	The write capacity for the table
Global Secondary Index Name	`No`	Name for the Global Secondary Index to update. If an index name is provided, **Global Secondary Index Read Capacity Units** and **Global Secondary Index Write Capacity Units** must be supplied.
Global Secondary Index Read Capacity Units	`No`	The read capacity for the global secondary index
Global Secondary Index Write Capacity Units	`No`	The write capacity for the global secondary index

Amazon DynamoDB Set Capacity Action

The `resource selector` AWS Lambda function searches the Amazon DynamoDB tables for tagged resources with the `Target Tag` in each region and account. However, there is no recommendable event pattern for this action.

Appendix C: Action Customization

Operations Conductor enables you to extend the solution by adding your own actions in the web consoles action catalog.

Create an Automation Document

Operations Conductor leverages AWS Systems Manager to perform actions on specific resources. Before adding a new action to the Systems Manager console, you must first create a new automation document. For more information, see Working with Automation Documents.

> **Note**
> We recommend first creating a new automation document in a text editor and then uploading the entire document to the Systems Manager console. Note that the document must be written in YAML format.

The following procedures show how to create an automation document that stops a running Amazon Elastic Compute Cloud (Amazon EC2) instance.

Document Description

This description will appear when viewing your document in the Systems Manager console and when viewing the action in the solution's web console.

```
description: <(Operations Conductor) Stops an EC2 Instance>
```

Schema Version

Currently, AWS Systems Manager automation documents only support Schema Version 0.3.

```
schemaVersion: 0.3
```

Operations Conductor Role

When Systems Manager executes your automation document, it assumes the role you provided. Replace the value of **assumeRole** with the ARN of the role that was created when the solution was launched. You can find your created ARN by navigating to the stack **Outputs** section of the template.

```
assumeRole: "<%%OperationsConductorSharedRoleArn%%>"
```

Parameters

The automation document requires the `SQSMsgBody`, and `SQSMsgReceiptHandle` parameters to run the document.

The `TargetResourceType` parameter determines what type of resources will be selected when the `resources selector` AWS Lambda function searches for the resources with the supplied `Target`

`Tag`. The parameter requires a value and must be formatted `service:resourceType`. In the below example, the parameter is set to `ec2:instance`.

The listed parameters above won't be shown in the web console. However, any additional parameters added will be shown in the UI, in order to be set during task creation. The following example shows an additional `ArbitraryParameter` which you can see in the web console and will be referenced later in the automation document.

```
parameters:
  SQSMsgBody:
    type: "String"
    description: "JSON Stringified version of the message body that was read off the
 Resource Queue"
  SQSMsgReceiptHandle:
    type: "String"
    description: "Receipt handle of the SQS message that was read off the queue"
  TargetResourceType:
    type: "String"
    description: "The AWS resource type for which this automation applies. The format of
 this value should be: service:resourceType"
    default: "ec2:instance"
  ArbitraryParameter:
    type: "String"
    description: "(Optional) An arbitrary value"
    default: ""
```

Automation Steps

When the solution executes the automation document, a series of steps are performed to validate that the resources are still tagged with the correct `Target Tag`, execute the action on the resource, and update the solution's Amazon DynamoDB tables.

The `VALIDATE_MSG_CONTENTS` step inspects the body of the message that was read from the `resource queue` and parses out the parameters needed to perform the action.

If a Python exception is raised in this script, the automation document stops processing and no further steps are executed. The script raises an exception if any required parameters are missing from the `SQSMsgBody`. An output object is populated as the step is executed and is returned when the it finishes. Properties of this object are mapped as output parameters for this step and will be referenced in subsequent steps.

The following code samples, show the individual steps of the automation.

```
mainSteps:
-   name: "VALIDATE_MSG_CONTENTS"
    action: aws:executeScript
    timeoutSeconds: 30
    description: "Validates contents of the message from the Resource Queue and parses out
 parameters"
    inputs:
      Runtime: python3.6
      Handler: script_handler
      InputPayload:
        SQSMsgBody: "{{ SQSMsgBody }}"
        ArbitraryParameter: "{{ ArbitraryParameter }}"
      Script: |-
        import boto3
        import json
        def script_handler(events, context):
          output = { "statusCode": 200 }
```

```
        sqs_msg_body = json.loads(events["SQSMsgBody"])

        if "ResourceId" not in sqs_msg_body:
          raise Exception("ResourceId was not found in the SQS Message Body.")

        output["resource_id"] = sqs_msg_body["ResourceId"]

        if "ResourceRegion" not in sqs_msg_body:
          raise Exception("ResourceRegion was not found in the SQS Message Body.")
        output["source_region"] = sqs_msg_body["ResourceRegion"]

        if "ResourceAccount" not in sqs_msg_body:
          raise Exception("ResourceAccount was not found in the SQS Message Body.")
        output["source_account_id"] = sqs_msg_body["ResourceAccount"]

        if "TargetTag" not in sqs_msg_body:
          raise Exception("TargetTag was not found in the SQS Message Body.")
        output["target_tag_name"] = sqs_msg_body["TargetTag"]

        if "TaskId" not in sqs_msg_body:
          raise Exception("TaskId was not found in the SQS Message Body.")
        output["task_id"] = sqs_msg_body["TaskId"]

        if "ParentExecutionId" not in sqs_msg_body:
          raise Exception("ParentExecutionId was not found in the SQS Message Body.")
        output["parent_execution_id"] = sqs_msg_body["ParentExecutionId"]

        output["arbitrary_parameter"] = events["ArbitraryParameter"]

        return output
    outputs:
    - Name: "resource_id"
      Selector: "$.Payload.resource_id"
      Type: "String"
    - Name: "source_region"
      Selector: "$.Payload.source_region"
      Type: "String"
    - Name: "source_account_id"
      Selector: "$.Payload.source_account_id"
      Type: "String"
    - Name: "target_tag_name"
      Selector: "$.Payload.target_tag_name"
      Type: "String"
    - Name: "task_id"
      Selector: "$.Payload.task_id"
      Type: "String"
    - Name: "parent_execution_id"
      Selector: "$.Payload.parent_execution_id"
      Type: "String"
    - Name: "arbitrary_parameter"
      Selector: "$.Payload.arbitrary_parameter"
      Type: "String"
```

The `CREATE_PERFORM_ACTION_AUTOMATION_EXECUTION_RECORD` step will log a record of the current document execution in the `automation executions` Amazon DynamoDB table.

The value for `parentExecutionId` is the output from the previous step and the value for `automationExecutionId` is provisioned by Systems Manager.

```
-   name: "CREATE_PERFORM_ACTION_AUTOMATION_EXECUTION_RECORD"
    action: aws:executeAwsApi
    timeoutSeconds: 30
    description: "Creates a record of this automation execution in the Operations Conductor
 Automation Executions Table"
```

```
    inputs: {
      "Service": "dynamodb",
      "Api": "PutItem",
      "TableName": "<%%AutomationExecutionsTableName%%>",
      "Item": {
        "parentExecutionId": { "S":
 "{{ SSM_BRANCH_EXECUTE_ACTION.parent_execution_id }}" },
        "automationExecutionId": { "S": "{{ automation:EXECUTION_ID }}" }
      }
    }
```

The `PERFORM_ACTION_ON_RESOURCE` step assumes the AWS Identity and Access Management (IAM) role that was generated for the task. Assuming the role enables the script to continue and perform the action on the resource.

In the first line of code, the script handler demonstrates how you would access the value of a parameter that was defined earlier in the automation document. For steps of `type aws:executeScript`, parameters must be defined in the `InputPayload` and accessed within an argument passed to the script handler.

```
-   name: "PERFORM_ACTION_ON_RESOURCE"
    action: aws:executeScript
    timeoutSeconds: 30
    description: "Stops an EC2 instance"
    inputs:
      Runtime: python3.6
      Handler: script_handler
      InputPayload:
        arbitrary_parameter: "{{ VALIDATE_MSG_CONTENTS.arbitrary_parameter }}"
        source_account_id: "{{ VALIDATE_MSG_CONTENTS.source_account_id }}"
        source_region: "{{ VALIDATE_MSG_CONTENTS.source_region }}"
        resource_id: "{{ VALIDATE_MSG_CONTENTS.resource_id }}"
        target_tag_name: "{{ VALIDATE_MSG_CONTENTS.target_tag_name }}"
        task_id: "{{ VALIDATE_MSG_CONTENTS.task_id}}"
      Script: |-
        import boto3
        import json
        def script_handler(events, context):
          print(f"Arbitrary Parameter Value: { events['arbitrary_parameter'] }")
          source_account_id = events["source_account_id"]
          source_region = events["source_region"]
          task_id = events["task_id"]

          # Assume role in source account
          sts_connection = boto3.client('sts')
          assumed_role = sts_connection.assume_role(
              RoleArn=f"arn:aws:iam::{source_account_id}:role/{source_account_id}-
{source_region}-{task_id}",
              RoleSessionName="ops_conductor_stop_instance"
          )

          # Look up the Instance by ID and make sure it is still tagged correctly
          ec2_client = boto3.client(
            'ec2',
            region_name=source_region,
            aws_access_key_id=assumed_role['Credentials']['AccessKeyId'],
            aws_secret_access_key=assumed_role['Credentials']['SecretAccessKey'],
            aws_session_token=assumed_role['Credentials']['SessionToken']
          )

          desc_instance_response =
 ec2_client.describe_instances(InstanceIds=[events["resource_id"]])

          instance = desc_instance_response["Reservations"][0]["Instances"][0]
```

```
          tag_found = False
          instance_tags = instance["Tags"]
          for tag in instance_tags:
            if tag["Key"] == events["target_tag_name"]:
              tag_found = True
              break

          if not tag_found:
            raise Exception(f"Instance ({ events['resource_id'] }) was found but it was not
 tagged with { events['target_tag_name'] }.")

          print(f"Instance ({ events['resource_id'] }) is still tagged with
 { events['target_tag_name'] }. Stopping the instance.")

          stop_params = {
            "InstanceIds": [events["resource_id"]]
          }

          stop_response = ec2_client.stop_instances(**stop_params)

          print("Success")
          print(f"{str(stop_response)}")

          return  { 'statusCode': 200 }
```

The `REMOVE_MSG_FROM_RESOURCE_QUEUE` step provides information of the action that was performed on the resource.

Verify that you replace *`%%ResourceQueueUrl%%`* with the URL of the **Resource Queue** that was created when the AWS CloudFormation template was deployed.

```
-   name: "REMOVE_MSG_FROM_RESOURCE_QUEUE"
    action: "aws:executeAwsApi"
    inputs: {
      "Service": "sqs",
      "Api": "DeleteMessage",
      "QueueUrl": "%%ResourceQueueUrl%%",
      "ReceiptHandle": "{{ SQSMsgReceiptHandle }}"
    }
```

The UPDATE_AUTOMATION_EXECUTION_RECORD step updates the record for the execution of this action on a resource. Progress is shown in the solution's web console.

Verify that you replace *`%% AutomationExecutionsTableName%%`* with the name of the `automation executions` Amazon DynamoDB table that was created when the AWS CloudFormation template was deployed.

```
-   name: "UPDATE_AUTOMATION_EXECUTION_RECORD"
    action: aws:executeAwsApi
    timeoutSeconds: 30
    description: "Updates the record of this automation execution in the Operations
 Conductor Automation Executions Table to mark it as successfully completed"
    inputs: {
      "Service": "dynamodb",
      "Api": "UpdateItem",
      "TableName": "<%%AutomationExecutionsTableName%%>",
      "Key": {
        "parentExecutionId": { "S": "{{ VALIDATE_MSG_CONTENTS.parent_execution_id }}" },
        "automationExecutionId": { "S": "{{ automation:EXECUTION_ID }}" }
      },
      "UpdateExpression": "SET #stat = :val1",
      "ExpressionAttributeNames": {
```

```
        "#stat": "status"
      },
      "ExpressionAttributeValues": {
        ":val1": { "S": "Success" }
      }
    }
```

The `UPDATE_TASK_EXECUTIONS_RECORD` step updates the record for the current execution of the task. Progress can be tracked in the solution's web console.

Verify that you replace *`%%TaskExecutionsTableName%%`* with the name of the `task executions` Amazon DynamoDB table that was created when the AWS CloudFormation template was deployed.

```
-   name: "UPDATE_TASK_EXECUTIONS_RECORD"
    action: aws:executeAwsApi
    timeoutSeconds: 30
    description: "Updates the record for the overall execution of the Operations Conductor
 Task that spawned this automation"
    inputs: {
      "Service": "dynamodb",
      "Api": "UpdateItem",
      "TableName": "<%%TaskExecutionsTableName%%>",
      "Key": {
        "taskId": { "S": "{{ VALIDATE_MSG_CONTENTS.task_id }}" },
        "parentExecutionId": { "S": "{{ VALIDATE_MSG_CONTENTS.parent_execution_id }}" }
      },
      "UpdateExpression": "SET completedResourceCount = completedResourceCount + :incr,
 lastUpdateTime = :uptime",
      "ExpressionAttributeValues": {
        ":incr": { "N": "1" },
        ":uptime": { "S": "{{ global:DATE_TIME }}" }
      }
    }
```

The `CHECK_FOR_TASK_EXECUTION_COMPLETION` step verifies that the number of resources successfully processed for this task matches the total number of resources found when the task was initially executed. If it matches, the status of the overall task is set to success, and will be shown in the web console.

Verify that you replace *`%%TaskExecutionsTableName%%`* with the name of the `task executions` Amazon DynamoDB table that was created when the AWS CloudFormation template was deployed.

You must include `onFailure: Continue`. This enables the automation document to continue successfully if the **ConditionExpression** of the **UpdateItem** call isn't met. This condition will only be met upon completion of the final resource for this task.

```
-   name: "CHECK_FOR_TASK_EXECUTION_COMPLETION"
    action: aws:executeAwsApi
    timeoutSeconds: 30
    description: "Marks the overall task execution as Success if all resources have been
 successfully acted on"
    inputs: {
      "Service": "dynamodb",
      "Api": "UpdateItem",
      "TableName": "<%%TaskExecutionsTableName%%>",
      "Key": {
        "taskId": { "S": "{{ VALIDATE_MSG_CONTENTS.task_id }}" },
        "parentExecutionId": { "S": "{{ VALIDATE_MSG_CONTENTS.parent_execution_id }}" }
      },
      "UpdateExpression": "SET #s = :stat",
      "ConditionExpression": "completedResourceCount = totalResourceCount",
      "ExpressionAttributeNames": {
```

```
        "#s": "status"
      },
      "ExpressionAttributeValues": {
        ":stat": { "S": "Success" }
      }
    }
    onFailure: Continue
    isEnd: true
```

Create IAM Role Template

You must define an AWS Identity and Access Management (IAM) role with the permissions required to perform the given action on the resources you specify. The solution's scripts will assume this role when performing and verifying the action. In the event of stopping an Amazon EC2 instance, the instance must have the `ec2:DescribeInstances`, `ec2:StopInstances`, and `tag:GetResources` permissions. Verify that you have made the following changes:

- *%%ResourceSelectorExecutionRoleArn%%*: Replace with the ARN of the Operations Conductor resource selector.
- *%%OperationsConductorSharedRoleArn%%*: Replace with the ARN of the Operations Conductor shared role.
- *%%MASTER_ACCOUNT%%*: The 12-digit ID of the account where you launched the solution's AWS CloudFormation template.

The name of the role will be set using the following format: *<AccountId>*-$*<Region>*-*<TaskId>*. **AccountId** and **Region** will be replaced by the AWS CloudFormation template, and must be set manually. Do not replace `%%TASK_ID%%`, the solution will automatically set the value when a task is created for this action.

```
AWSTemplateFormatVersion: "2010-09-09"
Description: "(SO0065) - Operations Conductor Stop Instances for cross accounts/regions."

Mappings:
    MasterAccount:
        ResourceSelectorExecutionRole:
            Name: "<%%ResourceSelectorExecutionRoleArn%%>"
        DocumentAssumeRole:
            Name: "<%%OperationsConductorSharedRoleArn%%>"
        Account:
            Id: "%%MASTER_ACCOUNT%%"

Resources:
    IAMRole:
        Type: AWS::IAM::Role
        Description: "Role to allow master account to perform actions"
        Properties:
            RoleName: !Sub "${AWS::AccountId}-${AWS::Region}-%%TASK_ID%%"
            AssumeRolePolicyDocument:
                Version: "2012-10-17"
                Statement:
                    -
                        Effect: "Allow"
                        Principal:
                            AWS:
                                - !FindInMap ["MasterAccount", "Account", "Id"]
                        Action:
                            - "sts:AssumeRole"
            Path: "/"
```

```
            Policies:
                -
                    PolicyName: "OperationsConductor-StopInstances"
                    PolicyDocument:
                        Version: "2012-10-17"
                        Statement:
                            -
                                Effect: "Allow"
                                Action:
                                    - "ec2:DescribeInstances"
                                    - "ec2:StopInstances"
                                    - "tag:GetResources"
                                Resource:
                                    - "*"
                            -
                                Effect: "Allow"
                                Action:
                                    - "iam:PassRole"
                                Resource:
                                    - !FindInMap ["MasterAccount",
"ResourceSelectorExecutionRole", "Name"]
                                    - !FindInMap ["MasterAccount", "DocumentAssumeRole",
"Name"]
```

Upload the Automation Document to Systems Manager

Use the following procedure to upload the automation document to AWS Systems Manager.

1. In the AWS Systems Manager console in your primary account, navigate to **Shared Resources**.
2. Select **Documents** on the left.
3. Select **Owned by me** to view the documents owned by this account.
4. Select **Create automation**.

   We recommend naming your document using the same format as the documents the solution created when the primary AWS CloudFormation template was launched. For example, *<solution-stack-name>*-OperationsConductor-*<action-name>*.
5. Select the **Editor** tab, then select **Edit**, and select **OK**.
6. In the **Document Editor**, paste the contents of the automation document, and select **Create automation**.
7. Select **Owned by me**, and navigate to the uploaded document.
8. Select **Details**, navigate to **Tags**, and select **Edit**.
9. Enter the **Tag Key** and **Value** you defined in the **Document Tag Key** and **Document Tag Value** template parameters when you launched the primary template.
10. Select **Save**.
11. Log in to the solution's web console, select **Get Started**, and verify that the new action is listed in the **Action Catalog**.

Upload the IAM Role Temaplate to Amazon S3

Use the following procedure to upload the IAM role template to the Amazon Simple Storage Service (Amazon S3) bucket.

Note
Verify that you are logged into the console in the primary account where you deployed the solution.

1. Navigate to the Amazon S3 console in your primary account.
2. Navigate to the solution-created Amazon S3 bucket, select **Create Folder**, and name the folder using the same name as the newly created automation document. Note that the solution requires you to use the same name for the folder and document.
3. Save the template as `cloudformation.template`, and upload the IAM role template.

Once completed, you can begin setting up tasks for your actions in the solution's web console.

Appendix D: Log Level

Operations Conductor enables you to change the log level of each microservice AWS Lambda function. To change the log level, you can change either the AWS CloudFormation stack parameter or each Lambda function's environment variable.

AWS CloudFormation Stack Parameter

To change the entire log level of microservice Lambda functions, you can update the CloudFormation stack parameter. When you change the **Log Level** parameter and update the AWS CloudFormation stack, it will automatically deploy the new log level to all AWS Lambda functions.

Lambda Function Environment Variable

Each Lambda function has a `LogLevel` environment variable. To change the value, you can change only specific microservice Lambda function's log level.

Supported Log Level

These are the supported log levels, and if you put other values, the microservice will change it to the default value, **2 (Info)**.

Parameters	Log Level	Description
0	Trace	The minimum log level is `trace`
1	Debug	The minimum log level is `debug`
2	Info	The minimum log level is `info`
3	Warn	The minimum log level is `warn`
4	Error	The minimum log level is `error`
5	Fatal	The minimum log level is `fatal`

Appendix E: Web Console

The solution creates a web console that enables users to create and configure tasks, monitor executions, and trigger manual task executions.

Web Console Menus

The solution's web console contains the following menus.

Tasks Menu

When logging in to the web console, this is the default menu that can be used to search, execute, create, edit, and delete tasks.

My Tasks

Shows created tasks.

Action Catalog

Shows the five actions which the solution supports. For more information on available actions, see Appendix B (p. 19).

Create a Task

This page enables you to configure tasks in detail. For more information about creating tasks, see Appendix A (p. 17).

Task Detail

Shows the task detail information. This page contains three tabs and four action buttons.

Tabs	Description
Overview	Shows detailed information about the task
Trigger	Shows the trigger information of the task
Logs	Shows the task execution logs • **Task Execution ID**: The task execution ID. When you select the task execution ID, you will be routed to the Task Automation Executions page. • **Status**: The status of the task execution. If a failure happens while executing the task with a resource, the status remains `InProgress`. • **Total Resources**: The number of total resources to be executed by the task execution. • **Completed Resources**: The number of completed resources. This number will be

Tabs	Description
	updated when the resource job is successfully completed by AWS Systems Manager. • **Start Time**: The start time of the task execution. You can sort the result based on the start time.

Action Buttons	Description
Edit Task	Enables you to update the task
Start Manual Task Execution	Manually starts the task. Note that this button will not be available for event type tasks.
Disable/Enable Automatic Task Execution	• For the schedule type task, it disables the Amazon CloudWatch Events rule, and the task cannot be executed automatically. • For the event type task, the resource selector will skip the Amazon SNS message sent by the `event forwarder` Lambda function in the secondary region and account.
Delete Task	Deletes the task. However, it will not delete the secondary CloudFormation stacks, you need to manually delete the stacks.

Task Automation Executions

List of the AWS Systems Manager automation execution IDs. Selecting an ID, opens the **Task Automation Executions** page.

Automation Execution Detail

Shows detailed information of the automation execution. When you select the ID of a step, it will show the detailed information of the step.

Users

The users menu enables you to invite users, edit user groups, and delete users. The solution supports the following two groups: Admin and Member. Only users in the Admin group can access to this menu.

Appendix F: Collection of Operational Metrics

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When enabled, the following information is collected and sent to AWS:

- **Solution ID:** The AWS solution identifier
- **Version:** The solution version
- **Unique ID (UUID):** Randomly generated, unique identifier for each solution deployment
- **Timestamp:** Data-collection timestamp
- **Event Type:** An identifier for what event occured
- **Event Data:** Details about the event

The table below shows the events that will result in the collection of operational metrics.

Event Type	Evevnt Data
Scheduled Task Created	• Task Name • Automation Document Name • Region Count (the number of regions the task will be executed in) • Account Count (the number of accounts the task will be executed in) • Schedule Type (Fixed Rate or Cron) • Schedule Details (Rate Type and Interval or Cron expression)
Event Task Created	• Task Name • Automation Document Name • Region Count (the number of regions the task will be executed in) • Account Count (the number of accounts the task will be executed in) • Event Service Name (the AWS service that triggers the event) • Event States (the resource state changes events that trigger the event)
Task Deleted	• Task Name
Task Executed Manually	• Task Name • Automation Document Name • Region Count (the number of regions the task will be executed in)

Event Type	Evevnt Data
Task Executed by Event	• Task Name • Automation Document Name • Region Count (the number of regions the task will be executed in)
Task Executed by Schedule	• Task Name • Automation Document Name • Region Count (the number of regions the task will be executed in)
Failed Task Executions	• Task Name • Automation Document Name

Note that AWS will own the data gathered via this survey. Data collection will be subject to the AWS Privacy Policy. To opt out of this feature, complete one of the following tasks: a) Modify the AWS CloudFormation template mapping section as follows:

```
Send:
    AnonymousUsage:
        Data: "Yes"
```

to

```
Send:
    AnonymousUsage:
        Data: "No"
```

b) After the solution has been launched, find the `User Service`, `Task Service`, and `Resource Selector` functions in the Lambda console, and set the **SendAnonymousUsageData** environment variable to `No`.

Source Code

You can visit our GitHub repository to download the templates and scripts for this solution, and to share your customizations with others.

Document Revisions

Date	Change	
November 2019	Initial release	
September 2021	Release version 1.0.1: Lambda Runtime Environment update to `Nodejs14.x`. For more details, refer to the CHANGELOG.md file	
December 2021	Release version 1.0.2: Bug fix. For more details, refer to the CHANGELOG.md file	

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Operations Conductor is licensed under the Apache License Version 2.0 available at http://https://www.apache.org/licenses/LICENSE-2.0

Contributors

- Beomseok Lee
- Eric Quinones
- Nikhil Reddy